



03/19/99

A

Atty. Docket No. 22074661-25541

THE ASSISTANT COMMISSIONER FOR PATENTS  
Washington, D.C. 20231  
Sir:

Transmitted herewith for filing is the patent application of Inventor(s) **Kevin M. Pinter and Donald L. Bolen**

Entitled: **SYSTEM FOR GENERATING OPTIMIZED COMPUTER DATA FIELD CONVERSION ROUTINES**

Enclosed are:

- ☒ 16 sheets of specification and 15 sheet(s) of drawing(s).
- ☒ An Assignment of the invention to Platinum technology IP, inc. and an Assignment Recordation cover sheet.
- ☐ A certified copy of a priority application.
- ☐ A verified statement to establish small entity status under 37 C.F.R. §§ 1.9 and 1.27.
- ☒ An executed declaration/power of attorney.
- ☐ An Information Disclosure Statement and form PTO-1449.
- ☐ Other

The filing fee has been calculated as shown below:

	(Col. 1)	(Col. 2)		SMALL	ENTITY		OTHER THAN A SMALL ENTITY	
FOR:	NO. FILED	NO. EXTRA		RATE	FEE		RATE	FEE
BASIC FEE					\$380	<u>OR</u>		\$760
TOTAL CLAIMS	20- 20 =	* 0		x 9	\$	<u>OR</u>	x 18	\$ 0
INDEP CLAIMS	3- 3 =	* 0		x 39	\$	<u>OR</u>	x 78	\$ 0
MULTIPLE DEPENDENT CLAIM PRESENTED				x 130	\$	<u>OR</u>	x 260	\$
* If the difference in Col. 1 is less than zero, enter "0" in Col. 2				TOTAL	\$	<u>OR</u>	TOTAL	\$760

☒ Please charge Deposit Account No. **02-0393** in the amount of **\$760.00** to cover the filing fee. The Commissioner is also hereby authorized to charge payment of the following fees associated with this communication or credit any overpayment to Deposit Account No. **02-0393**. A duplicate copy of this sheet is enclosed.

- ☒ Any additional filing fees required under 37 C.F.R. § 1.16.
- ☒ Any patent application processing fees under 37 C.F.R. § 1.17.

☐ A check to cover the filing fee is enclosed.

☒ The Commissioner is hereby authorized to charge payment of the following fees during the pendency of this application or credit any overpayment to Deposit Account No. **02-0393**. A duplicate copy of this sheet is enclosed.

- ☒ Any patent application processing fees under 37 C.F.R. § 1.17.
- ☐ The issue fee set in 37 C.F.R. § 1.18 at or before mailing of the Notice of Allowance, pursuant to 37 C.F.R. § 1.311(b).
- ☒ Any filing fees under 37 C.F.R. § 1.16 for presentation of extra claims.

Date: March 19, 1999

**EXPRESS MAIL CERT. NO. EI827994554US**  
**DATE OF DEPOSIT: March 19, 1999**

Respectfully Submitted,

Chris Kolefas (Reg. No. 35,226)  
BAKER & MCKENZIE  
805 THIRD AVENUE  
NEW YORK, N.Y. 10022

(212)751-5700

JC542 U.S. PTO

09/273149



03/19/99

OFFICE OF THE ASSISTANT COMMISSIONER FOR PATENTS

[illegible]

## 15

In many instances during computer processing of information, computer data must be converted from one data field type to another. For example, whenever data is passed from one program to another, the data typically goes through several conversions during the process, such as converting from text digits to a binary number.

5 The typical technique for converting data includes using a generic data conversion routine. When an entire record of data must be converted, the conversion routine must determine what the characteristics or attributes are for each of the data fields in the record. This may require the conversion routine to execute the same decision tree for each field for each record even though each field has known characteristics that do not change on a row by row basis. Therefore, many computer cycles are wasted by asking questions such as "Is this field of type character, integer, etc.?" over and over for each data field.

Based on the foregoing, there is a need for a system that provides efficient conversion of data fields.

#### SUMMARY OF THE INVENTION

15 One embodiment of the present invention is a system for converting data from input field types to output field types. The system receives a plurality of input attributes and output attributes from an application program, dynamically generates a plurality of data field conversion routines for each set of input attributes and output attributes, and stores the plurality of data field conversion routines in memory that is accessible to the application program.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram that illustrates an overview of the functionality of an optimized conversion generator system in accordance with one embodiment of the present invention.

Fig. 2 is a flowchart of the steps performed by the system in accordance with one embodiment of the present invention to generate optimized conversion routines.

Fig. 3 is a flowchart of the steps executed by the application when using the routines to convert input fields to output fields.

Fig. 4 is a flowchart of the code generating steps executed the conversion generator system to generate code when called by the application.

Figs. 5a and 5b illustrate a general example of dynamic code building that is used in one embodiment of the present invention.

Figs. 6a - 6h illustrate a specific example of a dynamic code generation routine that performs CHARACTER to CHARACTER conversions.

DETAILED DESCRIPTION

One embodiment of the present invention is a system that generates optimized data field to data field conversion routines for each type of conversion required by an application program. Fig. 1 is a block diagram that illustrates an overview of the functionality of an optimized conversion generator system 20 in accordance with one

embodiment of the present invention. System 20 can be implemented in software and executed on a general purpose computer that includes a central processing unit, and memory. In one embodiment, system 20 is implemented with IBM/360 machine instructions.

5           An application program 10 requires one or more types of field conversions to be executed. For each type of conversion, application 10 provides to system 20 the input (or "source") and output (or "destination") field attributes. For each set of input and output field attributes, system 20 dynamically generates an optimized conversion routine 30 that performs the conversion. The optimized routines 30 are placed in storage that is available to application 10.

10           The routines 30 in one embodiment are generated as stand-alone routines that are capable of being serially reusable and are called by application 10 using, for example, an application program interface ("API") when a conversion is required. In another embodiment, the routines 30 are generated as code chunks that are inserted  
15 inline with application 10 and are directly accessed when a conversion is required.

          One benefit of the present invention is that by building optimized conversion routines specifically tailored to the input and output field attributes, every execution of the routine saves numerous instructions that would normally be needed to identify field attributes each time the conversion is executed.

Fig. 2 is a flowchart of the steps performed by system 20 in accordance with one embodiment of the present invention to generate optimized conversion routines 30. The steps are executed by system 20 after application 10 determines at step 100 what attributes the input fields and output fields have.

5           At step 102, system 20 sets up the default process options of the generated conversion routines 30. The options may include whether the generated conversion routines 30 will be callable functions (i.e., able to be called by application 10), or copied inline into application 10. Step 102 builds a template interface block 104 which is an interface between application 10 and conversion generator system 20. Step 102 also generates an initiation call 106 that obtains the necessary storage and checks for errors.

          At step 108, a loop is initiated. The loop continues until all fields that must be converted are exhausted.

          Within the loop, at step 110 each set of input and output field attributes is received from application 10. The attributes are received through an API, and step 15           110 also builds a common field conversion interface block 116 based on the attributes.

          At step 112, the code generator of system 20 is called, using the common interface block 116. Step 112 generates code 118.

          At step 114, a function pointer that points to the generated field conversion routine 30 is saved.  
20

Fig. 3 is a flowchart of the steps executed by application 10 when using routines 30 to convert input fields to output fields.

During step 122, the application is processing. At step 124, the application obtains source or input data to convert. Typically, step 124 involves reading one or more records.

At step 126, a loop is initiated for each record read. At step 128, in one embodiment the appropriate conversion routine 30 for the conversion is called.

When all the data field and records are converted, at step 132 the code generator system 20 is called for termination. This results in freeing up memory at step 134.

At step 136, application 10 continues to process. Finally, at step 138 application 10 is completed.

Fig. 4 is a flowchart of the code generating steps executed by conversion generator system 20 to generate code when called by application 10.

At step 200, system 20 initializes by, for example, establishing the required storage, checking for invalid options, and specifying how the code should be generated.

At step 202, system 20 validates specific field conversion options such as verifying that the input and output lengths are correct. Step 202 also determines how

big the code will be when generated. This can be used by application 10 if the generated code will be stored inline.

At step 204, system 20 builds the conversion routine using field conversion interface block 116.

5 At step 206, the storage obtained at step 200 is released.

Steps 202 and 204 go through the same internal process. Therefore, at step 208 the input field type is determined. Examples of input field types include character input 210 or special time format input 212. However, any input field type is supported by the present invention.

Similarly, at step 214 the output field type is determined. Examples of output field types also include character input 213 or special time format input 215, but any output field type is supported by the present invention.

At step 216, if step 202 was executed, the size of the generated code is determined. At step 218, if step 204 was executed, the field conversion routines 30 are generated.

As disclosed, system 20 in accordance with one embodiment of the present invention dynamically generates optimized conversion routines 30 for each set of input and output field attributes. Routines 30 are then utilized by application 10 to process conversions. Input and output fields are categorized into archetypal data types by system 20, each with definable attributes and conversion behaviors. For example:



- Character data types will be a fixed length field with a maximum length attribute and a CCSID (or character set code page) attribute.
- Date data types will be a fixed length field with a maximum length attribute and a format attribute (ISO, EUR, etc) which determines location and type of separators used in date.

Some previously described or additional features included in one embodiment of optimized conversion generator system 20 include:

- Optionally obtain and free storage for API control blocks and/or generated code.
- API control blocks can be chained and templated by API management functions.
- API control blocks can be built through use of a macro interface.
- Conversion routines can utilize registers to address the input and output field locations directly. The registers can be chosen by application 10 through API parameters.
- The source field address register may optionally be incremented to the end of the input field after conversion based on API parameters.
- The target field address register may optionally be incremented to the end of the formatted field after conversion based on API parameters.

- An additional register may be incremented by the length of the converted field based on API parameters.
- Standard Linkage may be generated for conversion routines based on API parameters.
- Conversion Error exits may be specified to handle enumerated conversion error conditions based on API parameters.
- Character Code Set translation conversion code can be generated based on API parameters (i.e., ASCII character fields can be translated to EBCDIC character fields).
- Conversion routines can be generated to utilize the latest instructions supported by the level of the operating system for which the code is being generated.

In one embodiment, system 20 dynamically generates code by building code chunks in storage accessible by calling application 10 based on various settings in the API control block. Generating the code involves the following steps, as discussed in conjunction with the flowcharts:

1. Obtain storage for the code.
2. Identify code templates needed.
3. Move code templates.
4. Modify code templates.

5. Return executable code to calling application.

Further, in one embodiment system 20 can optionally, based on the API specification, generate program debugging instrumentation for the dynamically generated code. This instrumentation can include an optional dynamically allocated output file containing, for each field conversion: a report of the API options used for each dynamically generated routine that can be used to insure correctness of field attributes and general processing options; and a disassembled listing of the dynamically generated routine provided by an internal disassembler within system 20 that can be used to identify conversion code inaccuracies and areas of further optimization, and to help resolve generated code failures.

Figs. 5a and 5b illustrate a general example of dynamic code building that is used in one embodiment of the present invention.

Figs. 6a - 6h illustrate a specific example of a dynamic code generation routine that performs CHARACTER to CHARACTER conversions.

Several embodiments of the present invention are specifically illustrated and/or described herein. However, it will be appreciated that modifications and variations of the present invention are covered by the above teachings and within the purview of the appended claims without departing from the spirit and intended scope of the invention.

**WHAT IS CLAIMED IS:**

1           1. A method of converting a plurality of input field types to a plurality of  
2 output field types by an application program, said method comprising:

3           (a) receiving a first attribute of a first input field type and a second attribute of  
4 a first output field type;

5           (b) generating a first optimized conversion routine based on said first attribute  
6 and said second attribute; and

7           (c) executing said first optimized conversion routine from said application  
8 program to convert said first input field type to said first output field type.

9           2. The method of claim 1, wherein step (c) comprises calling said first  
10 optimized conversion routine from said application.

11           3. The method of claim 1, wherein step (c) comprises storing said first  
12 optimized conversion routine inline with said application.

1           4. The method of claim 1, wherein step (b) is performed dynamically while  
2 said application program is executing.

1           5. The method of claim 1, further comprising:

2 (d) receiving a third attribute of a second input field type and a fourth attribute  
3 of a second output field type;

4 (e) generating a second optimized conversion routine based on said third  
5 attribute and said fourth attribute; and

6 (f) executing said second optimized conversion routine from said application  
7 program to convert said second input field type to said second output field type.

6. The method of claim 1, wherein said first and second attribute is character  
type.

7. The method of claim 1, further comprising generating program debugging  
instrumentation for said first optimized conversion routine.

8. A method of converting data from input field types to output field types,  
said method comprising:

2 (a) receiving a plurality of input attributes and output attributes from an  
3 application program;

4 (b) dynamically generating a plurality of data field conversion routines for  
5 each set of input attributes and output attributes; and  
6

7 (c) storing said plurality of data field conversion routines in memory accessible  
8 to said application program.

1 9. The method of claim 8, wherein said data field conversion routines are  
2 callable by said application program.

1 10. The method of claim 8, wherein said data field conversion routines are  
stored inline said application program.

11. The method of claim 8, wherein step (b) is performed dynamically while  
said application program is executing.

12. The method of claim 8, wherein said input and output attributes are  
character type.

1 13. The method of claim 8, wherein said input and output attributes are date  
2 type.

1 14. The method of claim 8, further comprising generating program debugging  
2 instrumentation for said plurality of data field conversion routines.

1           15. A system for dynamically generating computer data field conversion  
2 routines, said system comprising:  
3           a processor; and  
4           a memory device coupled to said processor;  
5           wherein said system is adapted to receive a plurality of input attributes and  
6 output attributes from an application program; and  
7           wherein said memory device stores instructions that, when executed by said  
8 processor, cause said processor to:  
9           dynamically generate a plurality of data field conversion routines for each set  
10 of input attributes and output attributes; and  
11           store said plurality of data field conversion routines in a second memory  
12 device accessible to said application program.

13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57  
58  
59  
60  
61  
62  
63  
64  
65  
66  
67  
68  
69  
70  
71  
72  
73  
74  
75  
76  
77  
78  
79  
80  
81  
82  
83  
84  
85  
86  
87  
88  
89  
90  
91  
92  
93  
94  
95  
96  
97  
98  
99  
100  
101  
102  
103  
104  
105  
106  
107  
108  
109  
110  
111  
112  
113  
114  
115  
116  
117  
118  
119  
120  
121  
122  
123  
124  
125  
126  
127  
128  
129  
130  
131  
132  
133  
134  
135  
136  
137  
138  
139  
140  
141  
142  
143  
144  
145  
146  
147  
148  
149  
150  
151  
152  
153  
154  
155  
156  
157  
158  
159  
160  
161  
162  
163  
164  
165  
166  
167  
168  
169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179  
180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235  
236  
237  
238  
239  
240  
241  
242  
243  
244  
245  
246  
247  
248  
249  
250  
251  
252  
253  
254  
255  
256  
257  
258  
259  
260  
261  
262  
263  
264  
265  
266  
267  
268  
269  
270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325  
326  
327  
328  
329  
330  
331  
332  
333  
334  
335  
336  
337  
338  
339  
340  
341  
342  
343  
344  
345  
346  
347  
348  
349  
350  
351  
352  
353  
354  
355  
356  
357  
358  
359  
360  
361  
362  
363  
364  
365  
366  
367  
368  
369  
370  
371  
372  
373  
374  
375  
376  
377  
378  
379  
380  
381  
382  
383  
384  
385  
386  
387  
388  
389  
390  
391  
392  
393  
394  
395  
396  
397  
398  
399  
400  
401  
402  
403  
404  
405  
406  
407  
408  
409  
410  
411  
412  
413  
414  
415  
416  
417  
418  
419  
420  
421  
422  
423  
424  
425  
426  
427  
428  
429  
430  
431  
432  
433  
434  
435  
436  
437  
438  
439  
440  
441  
442  
443  
444  
445  
446  
447  
448  
449  
450  
451  
452  
453  
454  
455  
456  
457  
458  
459  
460  
461  
462  
463  
464  
465  
466  
467  
468  
469  
470  
471  
472  
473  
474  
475  
476  
477  
478  
479  
480  
481  
482  
483  
484  
485  
486  
487  
488  
489  
490  
491  
492  
493  
494  
495  
496  
497  
498  
499  
500  
501  
502  
503  
504  
505  
506  
507  
508  
509  
510  
511  
512  
513  
514  
515  
516  
517  
518  
519  
520  
521  
522  
523  
524  
525  
526  
527  
528  
529  
530  
531  
532  
533  
534  
535  
536  
537  
538  
539  
540  
541  
542  
543  
544  
545  
546  
547  
548  
549  
550  
551  
552  
553  
554  
555  
556  
557  
558  
559  
560  
561  
562  
563  
564  
565  
566  
567  
568  
569  
570  
571  
572  
573  
574  
575  
576  
577  
578  
579  
580  
581  
582  
583  
584  
585  
586  
587  
588  
589  
590  
591  
592  
593  
594  
595  
596  
597  
598  
599  
600  
601  
602  
603  
604  
605  
606  
607  
608  
609  
610  
611  
612  
613  
614  
615  
616  
617  
618  
619  
620  
621  
622  
623  
624  
625  
626  
627  
628  
629  
630  
631  
632  
633  
634  
635  
636  
637  
638  
639  
640  
641  
642  
643  
644  
645  
646  
647  
648  
649  
650  
651  
652  
653  
654  
655  
656  
657  
658  
659  
660  
661  
662  
663  
664  
665  
666  
667  
668  
669  
670  
671  
672  
673  
674  
675  
676  
677  
678  
679  
680  
681  
682  
683  
684  
685  
686  
687  
688  
689  
690  
691  
692  
693  
694  
695  
696  
697  
698  
699  
700  
701  
702  
703  
704  
705  
706  
707  
708  
709  
710  
711  
712  
713  
714  
715  
716  
717  
718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773  
774  
775  
776  
777  
778  
779  
780  
781  
782  
783  
784  
785  
786  
787  
788  
789  
790  
791  
792  
793  
794  
795  
796  
797  
798  
799  
800  
801  
802  
803  
804  
805  
806  
807  
808  
809  
810  
811  
812  
813  
814  
815  
816  
817  
818  
819  
820  
821  
822  
823  
824  
825  
826  
827  
828  
829  
830  
831  
832  
833  
834  
835  
836  
837  
838  
839  
840  
841  
842  
843  
844  
845  
846  
847  
848  
849  
850  
851  
852  
853  
854  
855  
856  
857  
858  
859  
860  
861  
862  
863  
864  
865  
866  
867  
868  
869  
870  
871  
872  
873  
874  
875  
876  
877  
878  
879  
880  
881  
882  
883  
884  
885  
886  
887  
888  
889  
890  
891  
892  
893  
894  
895  
896  
897  
898  
899  
900  
901  
902  
903  
904  
905  
906  
907  
908  
909  
910  
911  
912  
913  
914  
915  
916  
917  
918  
919  
920  
921  
922  
923  
924  
925  
926  
927  
928  
929  
930  
931  
932  
933  
934  
935  
936  
937  
938  
939  
940  
941  
942  
943  
944  
945  
946  
947  
948  
949  
950  
951  
952  
953  
954  
955  
956  
957  
958  
959  
960  
961  
962  
963  
964  
965  
966  
967  
968  
969  
970  
971  
972  
973  
974  
975  
976  
977  
978  
979  
980  
981  
982  
983  
984  
985  
986  
987  
988  
989  
990  
991  
992  
993  
994  
995  
996  
997  
998  
999  
1000  
1001  
1002  
1003  
1004  
1005  
1006  
1007  
1008  
1009  
1010  
1011  
1012  
1013  
1014  
1015  
1016  
1017  
1018  
1019  
1020  
1021  
1022  
1023  
1024  
1025  
1026  
1027  
1028  
1029  
1030  
1031  
1032  
1033  
1034  
1035  
1036  
1037  
1038  
1039  
1040  
1041  
1042  
1043  
1044  
1045  
1046  
1047  
1048  
1049  
1050  
1051  
1052  
1053  
1054  
1055  
1056  
1057  
1058  
1059  
1060  
1061  
1062  
1063  
1064  
1065  
1066  
1067  
1068  
1069  
1070  
1071  
1072  
1073  
1074  
1075  
1076  
1077  
1078  
1079  
1080  
1081  
1082  
1083  
1084  
1085  
1086  
1087  
1088  
1089  
1090  
1091  
1092  
1093  
1094  
1095  
1096  
1097  
1098  
1099  
1100  
1101  
1102  
1103  
1104  
1105  
1106  
1107  
1108  
1109  
1110  
1111  
1112  
1113  
1114  
1115  
1116  
1117  
1118  
1119  
1120  
1121  
1122  
1123  
1124  
1125  
1126  
1127  
1128  
1129  
1130  
1131  
1132  
1133  
1134  
1135  
1136  
1137  
1138  
1139  
1140  
1141  
1142  
1143  
1144  
1145  
1146  
1147  
1148  
1149  
1150  
1151  
1152  
1153  
1154  
1155  
1156  
1157  
1158  
1159  
1160  
1161  
1162  
1163  
1164  
1165  
1166  
1167  
1168  
1169  
1170  
1171  
1172  
1173  
1174  
1175  
1176  
1177  
1178  
1179  
1180  
1181  
1182  
1183  
1184  
1185  
1186  
1187  
1188  
1189  
1190  
1191  
1192  
1193  
1194  
1195  
1196  
1197  
1198  
1199  
1200  
1201  
1202  
1203  
1204  
1205  
1206  
1207  
1208  
1209  
1210  
1211  
1212  
1213  
1214  
1215  
1216  
1217  
1218  
1219  
1220  
1221  
1222  
1223  
1224  
1225  
1226  
1227  
1228  
1229  
1230  
1231  
1232  
1233  
1234  
1235  
1236  
1237  
1238  
1239  
1240  
1241  
1242  
1243  
1244  
1245  
1246  
1247  
1248  
1249  
1250  
1251  
1252  
1253  
1254  
1255  
1256  
1257  
1258  
1259  
1260  
1261  
1262  
1263  
1264  
1265  
1266  
1267  
1268  
1269  
1270  
1271  
1272  
1273  
1274  
1275  
1276  
1277  
1278  
1279  
1280  
1281  
1282  
1283  
1284  
1285  
1286  
1287  
1288  
1289  
1290  
1291  
1292  
1293  
1294  
1295  
1296  
1297  
1298  
1299  
1300  
1301  
1302  
1303  
1304  
1305  
1306  
1307  
1308  
1309  
1310  
1311  
1312  
1313  
1314  
1315  
1316  
1317  
1318  
1319  
1320  
1321  
1322  
1323  
1324  
1325  
1326  
1327  
1328  
1329  
1330  
1331  
1332  
1333  
1334  
1335  
1336  
1337  
1338  
1339  
1340  
1341  
1342  
1343  
1344  
1345  
1346  
1347  
1348  
1349  
1350  
1351  
1352  
1353  
1354  
1355  
1356  
1357  
1358  
1359  
1360  
1361  
1362  
1363  
1364  
1365  
1366  
1367  
1368  
1369  
1370  
1371  
1372  
1373  
1374  
1375  
1376  
1377  
1378  
1379  
1380  
1381  
1382  
1383  
1384  
1385  
1386  
1387  
1388  
1389  
1390  
1391  
1392  
1393  
1394  
1395  
1396  
1397  
1398  
1399  
1400  
1401  
1402  
1403  
1404  
1405  
1406  
1407  
1408  
1409  
1410  
1411  
1412  
1413  
1414  
1415  
1416  
1417  
1418  
1419  
1420  
1421  
1422  
1423  
1424  
1425  
1426  
1427  
1428  
1429  
1430  
1431  
1432  
1433  
1434  
1435  
1436  
1437  
1438  
1439  
1440  
1441  
1442  
1443  
1444  
1445  
1446  
1447  
1448  
1449  
1450  
1451  
1452  
1453  
1454  
1455  
1456  
1457  
1458  
1459  
1460  
1461  
1462  
1463  
1464  
1465  
1466  
1467  
1468  
1469  
1470  
1471  
1472  
1473  
1474  
1475  
1476  
1477  
1478  
1479  
1480  
1481  
1482  
1483  
1484  
1485  
1486  
1487  
1488  
1489  
1490  
1491  
1492  
1493  
1494  
1495  
1496  
1497  
1498  
1499  
1500  
1501  
1502  
1503  
1504  
1505  
1506  
1507  
1508  
1509  
1510  
1511  
1512  
1513  
1514  
1515  
1516  
1517  
1518  
1519  
1520  
1521  
1522  
1523  
1524  
1525  
1526  
1527  
1528  
1529  
1530  
1531  
1532  
1533  
1534  
1535  
1536  
1537  
1538  
1539  
1540  
1541  
1542  
1543  
1544  
1545  
1546  
1547  
1548  
1549  
1550  
1551  
1552  
1553  
1554  
1555  
1556  
1557  
1558  
1559  
1560  
1561  
1562  
1563  
1564  
1565  
1566  
1567  
1568  
1569  
1570  
1571  
1572  
1573  
1574  
1575  
1576  
1577  
1578  
1579  
1580  
1581  
1582  
1583  
1584  
1585  
1586  
1587  
1588  
1589  
1590  
1591  
1592  
1593  
1594  
1595  
1596  
1597  
1598  
1599  
1600  
1601  
1602  
1603  
1604  
1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613  
1614  
1615  
1616  
1617  
1618  
1619  
1620  
1621  
1622  
1623  
1624  
1625  
1626  
1627  
1628  
1629  
1630  
1631  
1632  
1633  
1634  
1635  
1636  
1637  
1638  
1639  
1640  
1641  
1642  
1643  
1644  
1645  
1646  
1647  
1648  
1649  
1650  
1651  
1652  
1653  
1654  
1655  
1656  
1657  
1658  
1659  
1660  
1661  
1662  
1663  
1664  
1665  
1666  
1667  
1668  
1669  
1670  
1671  
1672  
1673  
1674  
1675  
1676  
1677  
1678  
1679  
1680  
1681  
1682  
1683  
1684  
1685  
1686  
1687  
1688  
1689  
1690  
1691  
1692  
1693  
1694  
1695  
1696  
1697  
1698  
1699  
1700  
1701  
1702  
1703  
1704  
1705  
1706  
1707  
1708  
1709  
1710  
1711  
1712  
1713  
1714  
1715  
1716  
1717  
1718  
1719  
1720  
1721  
1722  
1723  
1724  
1725  
1726  
1727  
1728  
1729  
1730  
1731  
1732  
1733  
1734  
1735  
1736  
1737  
1738  
1739  
1740  
1741  
1742  
1743  
1744  
1745  
1746  
1747  
1748  
1749  
1750  
1751  
1752  
1753  
1754  
1755  
1756  
1757  
1758  
1759  
1760  
1761  
1762  
1763  
1764  
1765  
1766  
1767  
1768  
1769  
1770  
1771  
1772  
1773  
1774  
1775  
1776  
1777  
1778  
1779  
1780  
1781  
1782  
1783  
1784  
1785  
1786  
1787  
1788  
1789  
1790  
1791  
1792  
1793  
1794  
1795  
1796  
1797  
1798  
1799  
1800  
1801  
1802  
1803  
1804  
1805  
1806  
1807  
1808  
1809  
1810  
1811  
1812  
1813  
1814  
1815  
1816  
1817  
1818  
1819  
1820  
1821  
1822  
1823  
1824  
1825  
1826  
1827  
1828  
1829  
1830  
1831  
1832  
1833  
1834  
1835  
1836  
1837  
1838  
1839  
1840  
1841  
1842  
1843  
1844  
1845  
1846  
1847  
1848  
1849  
1850  
1851  
1852  
1853  
1854  
1855  
1856  
1857  
1858  
1859  
1860  
1861  
1862  
1863  
1864  
1865  
1866  
1867  
1868  
1869  
1870  
1871  
1872  
1873  
1874  
1875  
1876  
1877  
1878  
1879  
1880  
1881  
1882  
1883  
1884  
1885  
1886  
1887  
1888  
1889  
1890  
1891  
1892  
1893  
1894  
1895  
1896  
1897  
1898  
1899  
1900  
1901  
1902  
1903  
1904  
1905  
1906  
1907  
1908  
1909  
1910  
1911  
1912  
1913  
1914  
1915  
1916  
1917  
1918  
1919  
1920  
1921  
1922  
1923  
1924  
1925  
1926  
1927  
1928  
1929  
1930  
1931  
1932  
1933  
1934  
1935  
1936  
1937  
1938  
1939  
1940  
1941  
1942  
1943  
1944  
1945  
1946  
1947  
1948  
1949  
1950  
1951  
1952  
1953  
1954  
1955  
1956  
1957  
1958  
1959  
1960  
1961  
1962  
1963  
1964  
1965  
1966  
1967  
1968  
1969  
1970  
1971  
1972  
1973  
1974  
1975  
1976  
1977  
1978  
1979  
1980  
1981  
1982  
1983  
1984  
1985  
1986  
1987  
1988  
1989  
1990  
1991  
1992  
1993  
1994  
1995  
1996  
1997  
1998  
1999  
2000  
2001  
2002  
2003  
2004  
2005  
2006  
2007  
2008  
2009  
2010  
2011  
2012  
2013  
2014  
2015  
2016  
2017  
2018  
2019  
2020  
2021  
2022  
2023  
2024  
2025  
2026  
2027  
2028  
2029  
2030  
2031  
2032  
2033  
2034  
2035  
2036  
2037  
2038  
2039  
2040  
2041  
2042  
2043  
2044  
2045  
2046  
2047  
2048  
2049  
2050  
2051  
2052  
2053  
2054  
2055  
2056  
2057  
2058  
2059  
2060  
2061  
2062  
2063  
2064  
2065  
2066  
2067  
2068  
2069  
2070  
2071  
2072  
2073  
2074  
2075  
2076  
2077  
2078  
2079  
2080  
2081  
2082  
2083  
2084  
2085  
2086  
2087  
2088  
2089  
2090  
2091  
2092  
2093  
2094  
2095  
2096  
2097  
2098  
2099  
2100  
2101  
2102  
2103  
2104  
2105  
2106  
2107  
2108  
2109  
2110  
2111  
2112  
2113  
2114  
2115  
2116  
2117  
2118  
2119  
2120  
2121  
2122  
2123  
2124  
2125  
2126  
2127  
2128  
2129  
2130  
2131  
2132  
2133  
2134  
2135  
2136  
2137  
2138  
2139  
2140  
2141  
2142  
2143  
2144  
2145  
2146  
2147  
2148  
2149  
2150  
2151  
2152  
2153  
2154  
2155  
2156  
2157  
2158  
2159  
2160  
2161  
2162  
2163  
2164  
2165  
2166  
2167  
2168  
2169  
2170  
2171  
2172  
2173  
2174  
2175  
2176  
2177  
2178  
2179  
2180  
2181  
2182  
2183  
2184  
2185  
2186  
2187  
2188  
2189  
2190  
2191  
2192  
2193  
2194  
2195  
2196  
2197  
2198  
2199  
2200  
2201  
2202  
2203  
2204  
2205  
2206  
2207  
2208

1           18. The system of claim 15, wherein said plurality of data field conversion  
2 routines are generated while said application program is executing.

1           19. The system of claim 15, wherein said input attributes are character type  
2 and said output attributes are date type.

1           20. The system of claim 15, wherein said processor further generates program  
debugging instrumentation for said plurality of data field conversion routines.



ABSTRACT OF THE DISCLOSURE

A system converts data from input field types to output field types. The system receives a plurality of input attributes and output attributes from an application program, dynamically generates a plurality of data field conversion routines for each set of input attributes and output attributes, and stores the plurality of data field conversion routines in memory that is accessible to the application program.

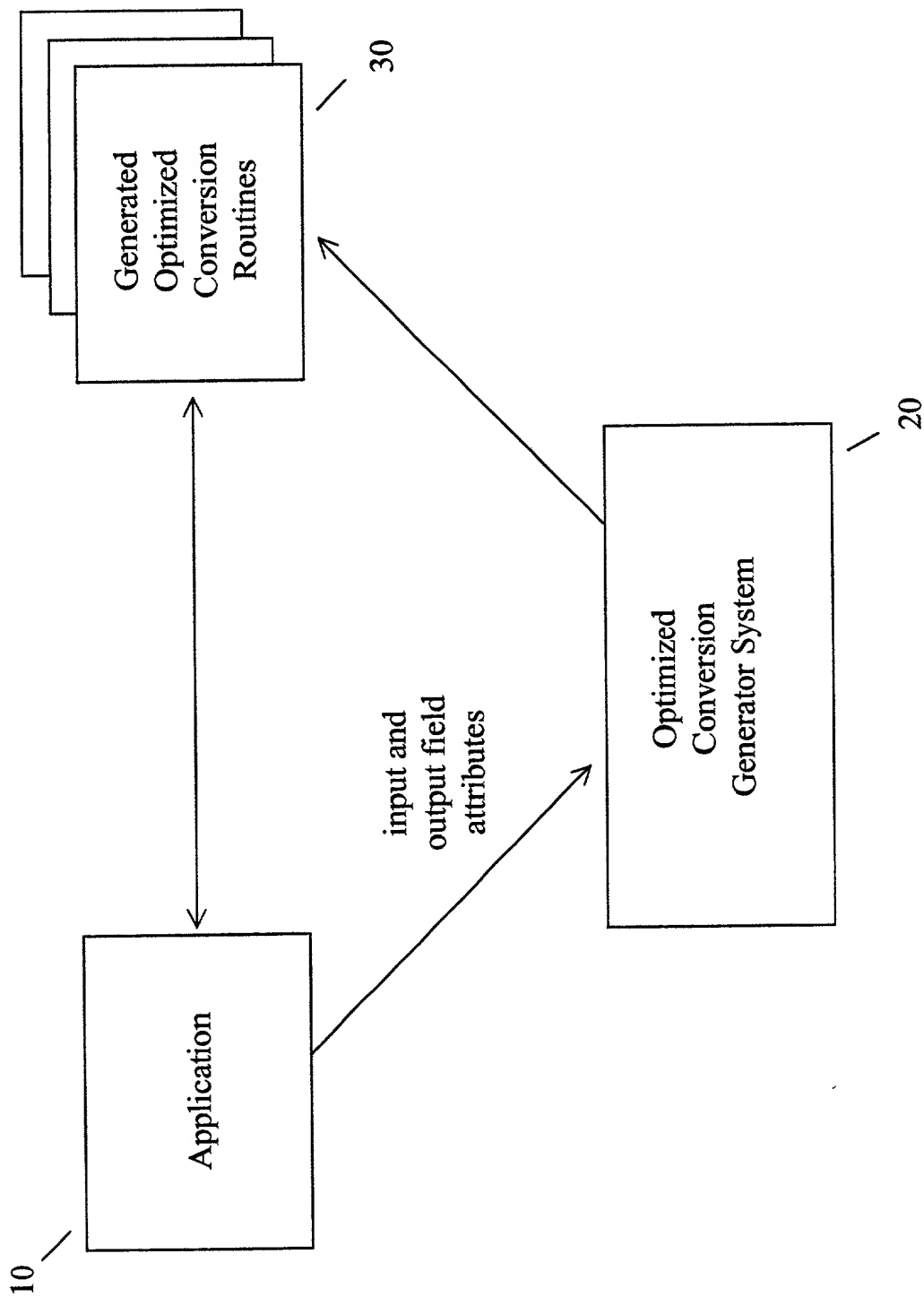


Fig. 1

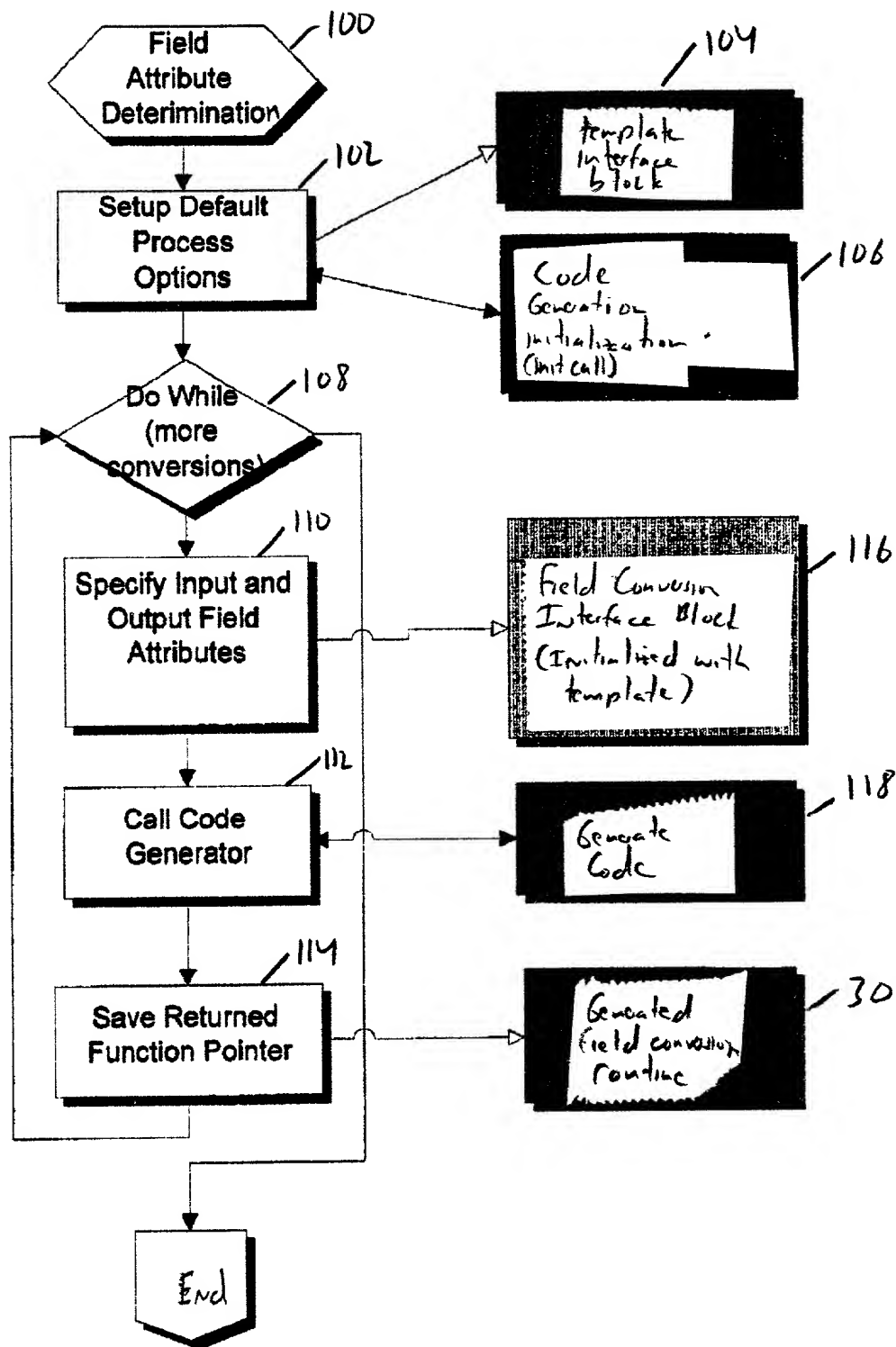


Fig. 2

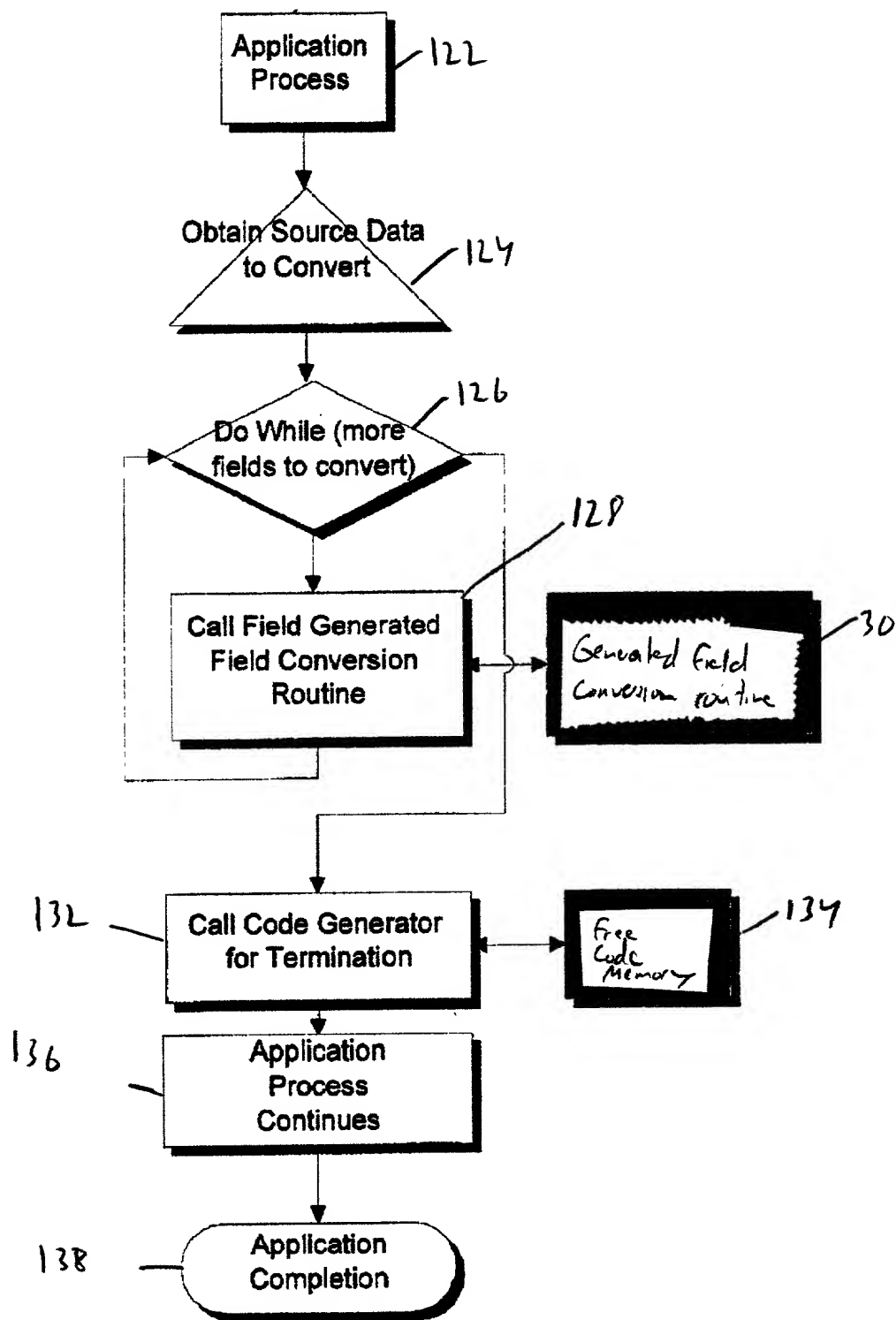


Fig. 3

Code Generation  
Package

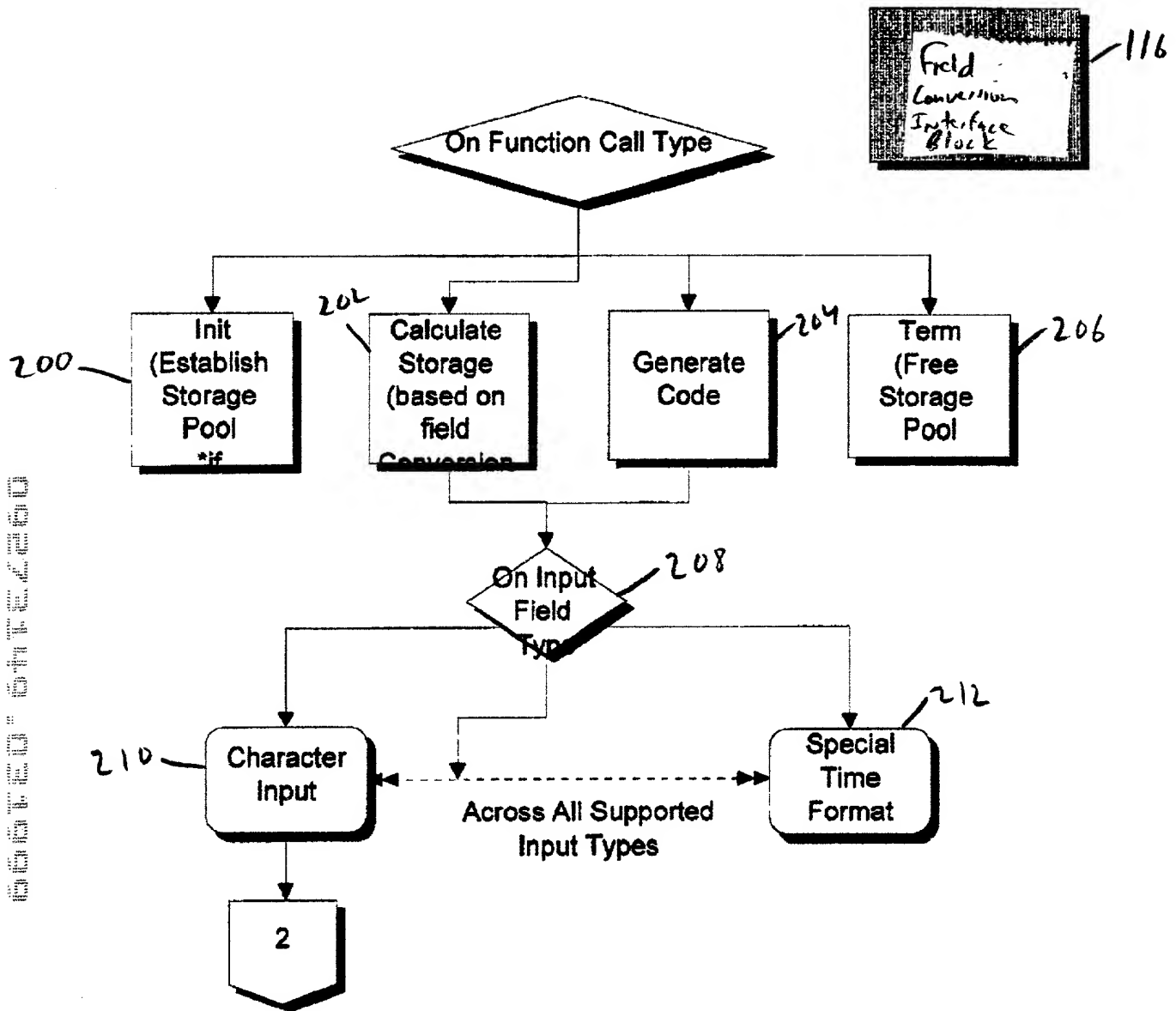


Fig. 4a

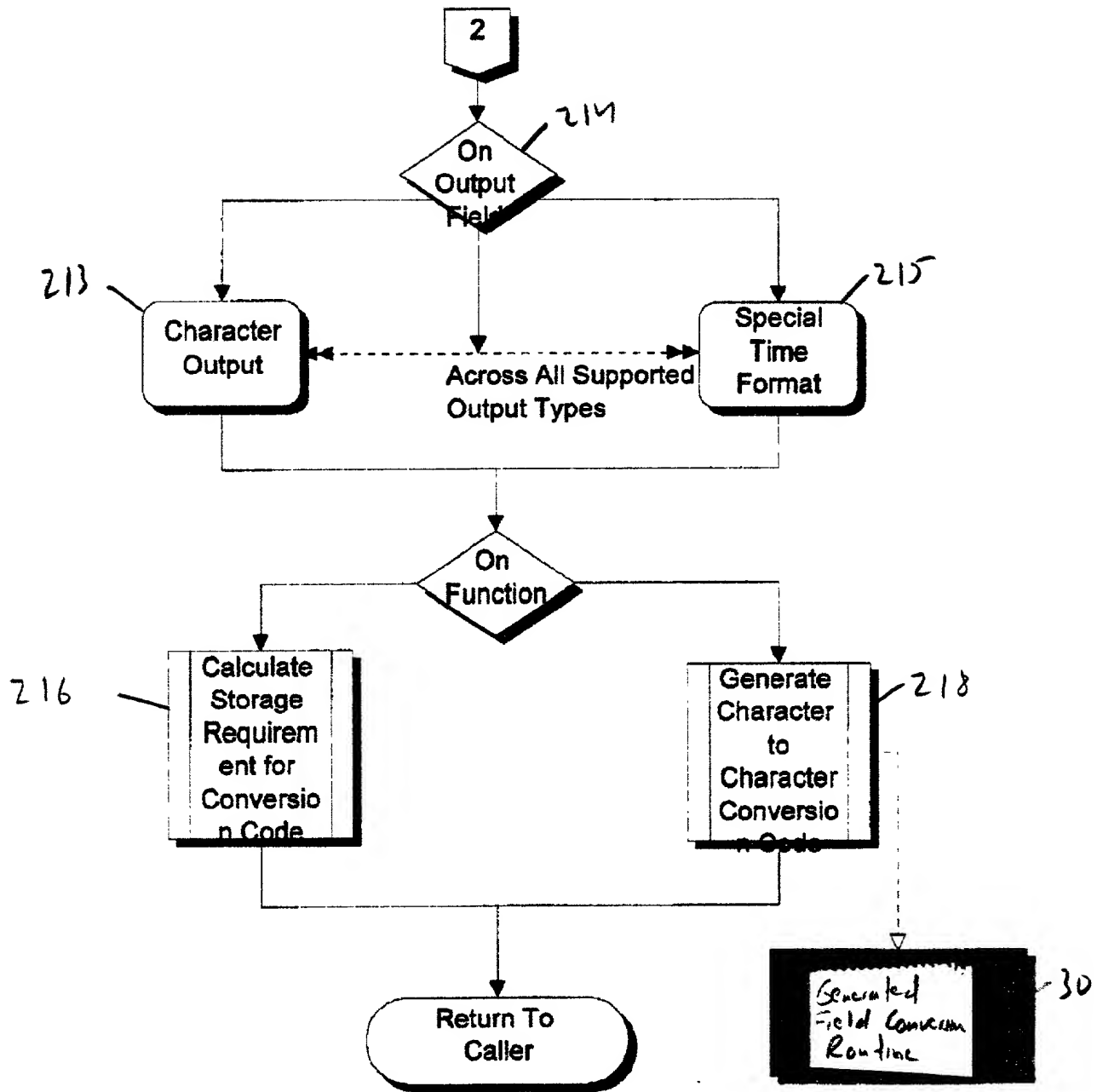


Fig. 4b

```

R5      =      Current Instruction Offset within application buffer
R6      =      Current Instruction Address within application buffer
R7      =      Work Register - used for calculating offsets, etc
R12     =      Base register of code generator and template code

```

```

SLR      R5,R5              clear offset
L        R6,$BCB_BCODE_@    get address of user buffer

```

```

* if linkage required call standard linkage builder

```

```

IF (TM,$BCB_PFLAG1,$BCB_LINKAGE,0)
    SETF    LINKAGE
    IF (CLI,$BCB_LINKAGE_TYPE,EQ,C'N')
        RESETF LINKAGE
    COND ELSE

```

```

* call standard linkage builder
    #BAS    14,=A(BURST_ENTRY_LINKAGE)
ENDIF

```

```

ELSE
    RESETF LINKAGE
ENDIF

```

```

****
STDRETURN      -      RETURN TO APPLICATION
* $BCB_BCODE_@ WILL POINT TO BUILT CODE
****

```

```

*
* Routine to build standard entry linkage
*
BURST_ENTRY_LINKAGE CSMSUBI BASE=R10,WORKREG=R3
*
* Move Template code into user buffer
    MVC     0(STD_ENTL_010_L,R6),STD_ENTL_010
*
* Modify " LA      R14,0(0)" instruction
* Get Offset to Savearea using equate STD_ENTL_010_SA_A
* Set base register for instruction to R12
* Set D(X,B) of instruction (R7 contains constructed D(X,B))
    LA      R7,STD_ENTL_010_SA_A(,R5)
    O       R7,=X'0000C000'
    STH     R7,STD_ENTL_010_SA_T(,R6)
*

```

**Fig. 5a**

```

* Modify " B      0(R12)" instruction
* Get offset of branch target using equate STD_ENTL_010_B_A_T
* Set D(X,B) of instruction (R7 contains constructed D(X,B))
* ** Note X (index register) has been set by assembler as R12
*   STH does not change the instruction's index register
      LA    R7,STD_ENTL_010_B_A_T(,R5)  CALC OFFSET FOR BRANCH TARGET
      STH   R7,STD_ENTL_010_B_A(,R6)    SET BRANCH D(X,B)
*
* Increment Next Instruction Offset (in R5) by length of code
* Increment Next Instruction Address (in R6) by length of code
      LA    R5,STD_ENTL_010_L(,R5)
      LA    R6,STD_ENTL_010_L(,R6)
*
* Return to caller
* Code has been built and the Instruction Offset and Address registers
* have been updated for next instruction construction

```

```

          CSMSUBO
*- STANDARD ENTRY LINKAGE -----
*
*-----
STD_ENTL_010 DS   OS
              STM    R14,R12,12(R13)
STD_ENTL_010_SA_T EQU  *-STD_ENTL_010+2
              LA     R14,0(0)          BURSTED SAVEAREA+0
              ST     R13,4(,R14)
              ST     R14,8(,R13)
              LR     R13,R14           R13 = BURSTED SAVEAREA
              LR     R12,R15           SET BURSTED BASE REG
STD_ENTL_010_B_A EQU  *-STD_ENTL_010+2
              B      0(R12)           WS_BRANCH
STD_ENTL_010_SA_A EQU  *-STD_ENTL_010
              DC     18F'0'
STD_ENTL_010_B_A_T EQU  *-STD_ENTL_010
STD_ENTL_010_L EQU    *-STD_ENTL_010
*-----

```

Fig. 5b



- \* Call made by API passing API \$BURSTCB control block
- \* Control block contains field attributes and conversion
- \* options
- \* Reset processing flags
- \* NO\_BUILD -> doing conversion routine storage calculation
- \* CALLED\_ROUTINE -> creating a called routine
- \* Check for API block -> if not there abend with dump
- \* Copy passed API block to working storage (IN\_BCB)

```

MAIN_0000 DS      OS
*
      RESETF  NO_BUILD
      RESETF  CALLED_ROUTINE
*
      LTR     R1,R1
      BNZ     MAIN_0005
*
      ABEND   001,DUMP
*
MAIN_0005 DS      OS
      MVC     IN_BCB($BCB_LENGTH),0(R1)
*
      LA      R9,IN_BCB          R9 = ADDRESS OF $BURSTCB
      USING   $BURSTCB,R9
*
* If calculate storage requested SET NO_BUILD
      IF (CLC,$BCB_FUNC,EQ,=Y($BCB_CALC_STORAGE))
          SETF  NO_BUILD
      ENDIF
*
* INITIALIZE WORKING STORAGE
* If actually BUILDING code (not NO_BUILD)
* 1. Obtain offset from beginning of BASE REGISTER
* for code. If callable routine this has been set to 0.
* otherwise this we are building inline code within the application's
* user managed buffer and the offset will set to current instruction offset
* within the buffer.
* 2. Obtain address of passed code buffer
* 3. Calculate current instruction address based on offset into buffer

```

**Fig. 6a**

```

MAIN_STRT DS      0S
      IF (-NO_BUILD)
        LH      R5,$BCB_BCODE_OFFSET
        L       R6,$BCB_BCODE_@
        LA      R6,0(R5,R6)
      ELSE
        SLR      R5,R5          CLEAR FOR ACCUM
        SLR      R6,R6          CLEAR FOR ACCUM
      ENDIF

*
* INITIALIZE WORK FIELDS FOR ANY COLUMN CONVERSION
* 1. Obtain input field's addressing register
* 2. Build RX type assembler instruction D(X,B) with offset 0
* 3. Obtain output field's addressing register
* 4. Build RX type assembler instruction D(X,B) with offset 0
*   set template for output D(X,B)
* 5. Obtain input and output lengths
* 6. Set Current working D(X,B) templates
      SLR      R7,R7
      ICM      R7,B'0001',$BCB_IREG
      SLL      R7,4              SHIFT NIBBLE
      STC      R7,WB_INIT_SOURCE_DB
      ICM      R7,B'0001',$BCB_OREG
      SLL      R7,4              SHIFT NIBBLE
      STC      R7,WB_INIT_TARGET_DB
      MVC      WB_TOT_INPUT_LEN,$BCB_ILEN
      MVC      WB_TOT_OUTPUT_LEN,$BCB_OLEN
      MVC      WB_SOURCE_DB,WB_INIT_SOURCE_DB    RESET DB
      MVC      WB_TARGET_DB,WB_INIT_TARGET_DB    RESET DB

*
* CHECK FOR LINKAGE REQUIREMENTS
* IF LINKAGE = E (BASIC ENTRY - SAVE/RESTORE R14) THEN
*   BURST_WORK_BRANCH WILL SAVE R14 AND SET RESTORE_R14
*   BURST_EXIT_LINKAGE RESTORES R14 AND BASR R14
* ENDIF
      RESETF   RESTORE_R14
      IF (TM,$BCB_PFLAG1,$BCB_LINKAGE,0)
        SETF    LINKAGE
        IF (CLI,$BCB_LINKAGE_TYPE,EQ,C'N')
          RESETF LINKAGE
        COND ELSE
          #BAS    14,=A(BURST_ENTRY_LINKAGE)
        ENDIF
      ELSE
        RESETF   LINKAGE
      ENDIF

```

Fig. 6b

```

* CALL INPUT TYPE PROCESSING ROUTINE
* 1. Get address of input field type table
*   This table contains an index of supported input types
*   with their associated code generation routines
* 2. Call code generation routine for Input field type
*   In this case INPUT FIELD TYPE IS CHARACTER
*   INPUT FIELD TYPE CHARACTER calls routine named CHARACTER
**** Further down subroutine CHARACTER is shown
      L      R14,=A(TYPE_TABLE)
      LH     R15,$BCB_ITYPE
      LA     R15,0(R14,R15)
      L      R15,0(R15)
      BASR   R14,R15

* Subroutine has built conversion code for INPUT TYPE CHARACTER and OUTPUT TYPE CHARACTER
* Check for other process options such as: accumulate a source addressing register,
* accumulate a target addressing register, or accumulate alternate register.
* alternate register usually is a total output length accumulator used by the calling
* application to keep track of an aggregate of all output field lengths
* 1. IF source addressing register accumulate requested build code to accumulate
* 2. IF target addressing register accumulate requested build code to accumulate
* 3. IF length register accumulate requested build code to accumulate
* 4. IF exit linkage requested build exit linkage
* 5. RETURN TO API CALLER with generated conversion routine
MAIN_0200 DS      OS
      IF (TM,$BCB_PFLAG1,$BCB_SRC_ACUM,0)
      LH     R0,WB_SOURCE_ACCUM_INDEX
      IC     R1,$BCB_SRC_ACUM_REG
      LH     R7,WB_TOT_INPUT_LEN
      #BAS   14,=A(FIXED_ACCUM)
      ENDIF
*
      IF (TM,$BCB_PFLAG1,$BCB_TRG_ACUM,0)
      LH     R0,WB_TARGET_ACCUM_INDEX
      IC     R1,$BCB_TRG_ACUM_REG
      LH     R7,WB_TOT_OUTPUT_LEN
      #BAS   14,=A(FIXED_ACCUM)
      ENDIF
*
      IF (TM,$BCB_PFLAG1,$BCB_TRG_L_ACUM,0)
      LH     R0,WB_TARGET_ACCUM_INDEX
      IC     R1,$BCB_TLN_ACUM_REG
      LH     R7,WB_TOT_OUTPUT_LEN
      #BAS   14,=A(FIXED_ACCUM)
      ENDIF
*
* BURST EXIT LINKAGE
      IF (LINKAGE)
      SETF   CLEAR_R15
      #BAS   14,=A(BURST_EXIT_LINKAGE)
      ENDIF
      RETURN to CALLER

```

Fig. 6c

```

*-----*
* Character Input Field Type Conversion Routine
* Abstract:
*   This routine is called to either build Character Input
*   Fields to all supported Output Field Types, or to calculate
*   storage requirements for generated conversion routines for
*   Input field type Character
*
* CHARACTER field type constraints
*   These field types will be of fixed length
*   Maximum length is 254 8bit bytes
*   They may be proceeded with a null field indicator of length
*   1 byte that will contain values of x'00' for non-null fields
*   and x'ff' for nulled fields. Nulled fields will not be
*   converted except to indicate on output that field was null
*   There values are of EBCDIC CCSID (character code set) unless
*   a CCSID is specified through the API.
*-----*

CHARACTER CSMSUBI BASE=R10,WORKREG=R3
* Use branch table generated by API to branch on output type (BTYP=0)
* Example is demonstrating character to character conversion
* Branch will be taken to CHAR_CHAR_0000
    L      R15,=A(RET_RC_32)
    $BURST  BTABLE,
            BREG=1,
            BTYP=0,
            UNSUPPORTED=0(,R15),
            CHAR=CHAR_CHAR_0000,
            LVARC=CHAR_VARC_0000,
            VARC=CHAR_VARC_0000
                                     X
                                     X
                                     X
                                     X
                                     X
                                     X

*-----*
* @PSEUDO-CODE@
*
* CHARACTER TO CHARACTER CONVERSION
*
* DETERMINE WORKING STORAGE
* Some conversions require the generation of local working storage
* Working storage is generated according to specific conversion options and
* specific input and output field attributes to avoid generating more storage
* than needed.
*
* IF CONVERTING CCSID'S (character code sets) THEN
*   IF using a character translation table (uses TR instruction)
*     Build BRANCH over working storage
*     Build FULL WORD to hold Address character translation table
*     UPDATE Previously built Branch instruction to branch to current offset
*     (offset is next halfword aligned byte where next instruction is to be built)
*   ENDIF
* ENDIF

```

Fig. 6d

```

* IF INPUT LENGTH is GREATER than OUTPUT LENGTH
* current implementation allows for truncation of trailing spaces
* If input field being converted by generated code contains non-spaces
* that won't fit into output field of lesser length then conversion
* error 4 routine will be called to return a value of 4 in R15
*
* 1. Build BRANCH over working storage
* 2. Build a buffer full of spaces to be used in INPUT field compare
* 3. Build Conversion error routine to return error #4
* 4. UPDATE Previously built Branch instruction to branch to current offset
* (offset is next halfword aligned byte where next instruction is to be built)
* ENDIF
* - DETERMINE WORKING STORAGE
*
* @PSEUDO-CODE@-----CHAR_CHAR_0000 DS 0S
*
* BURST WORKAREA IF CONVERSION ERROR OR CONVERT CCSID
* TM $BCB_PFLAG2,$BCB_CCSID_CNV
* BNZ CHAR_CHAR_0020
* CLC $BCB_ILEN,$BCB_OLEN
* BNH CHAR_CHAR_0040
*
CHAR_CHAR_0020 DS 0S
#BAS 14,=A(BURST_WORK_BRANCH)
*
IF (TM,$BCB_PFLAG2,$BCB_CCSID_CNV,NZ)
IF (TM,$BCB_PFLAG2,$BCB_CCSID_CNV_ATOE,0)
#BAS 14,=A(BURST_BWK_TO_E_XLATE_@)
ELSE
#BAS 14,=A(BURST_BWK_TO_O_XLATE_@)
ENDIF
#BAS 14,=A(BURST_BWK_FULL)
STH R7,WB_SAVE_R2_OFFSET
ENDIF
*
IF ILEN > OLEN THEN NEED FOLLOWING WORK FIELDS
* BURST BUFFER255 - SPACES
* BURST #@ERROR4 CALL
* ENDIF
IF (CLC,$BCB_ILEN,GT,$BCB_OLEN)
#BAS 14,=A(BURST_BWK_BUFFER255)
*
*
LA R1,4 #@ERROR4
#BAS 14,=A(BUILD_CNVERR)
ENDIF
*
#BAS 14,=A(UPDATE_WORK_BRANCH)
*
*

```

**Fig. 6e**

```

* IF OUTPUT NULLABLE THEN
*   BURST MOVEMENT OF NULL INDICATOR
*   R1 = X'00' FOR MVI Instruction Builder
*   WB_TARGET_DB (current target D(B)) USED FOR INDICATOR LOCATION
*   Build MVI OF NULL INDICATOR (MVI_0000)
*   UPDATE Current TARGET D(B) TO ALLOW DATA TO SKIP NULL INDICATOR
*   ADD 1 TO TOT OUTPUT LENGTH (FOR NULL INDC) (this allows for accumulation requests)
*   ENDIF
CHAR_CHAR_0040 DS 05
      IF (TM,$BCB_OFLAG1,$BCB_ONULL,0)
          SLR    R1,R1                CLEAR SOURCE BYTE
          #BAS   14,=A(MVI_0000)      BURST MVI NULL INDC
*
          LH     R1,WB_TARGET_DB       UPDATE TARGET DB
          LA     R1,1(,R1)
          STH    R1,WB_TARGET_DB
*
          LH     R1,WB_TOT_OUTPUT_LEN  UPDATE OUTPUT LEN
          LA     R1,1(,R1)
          STH    R1,WB_TOT_OUTPUT_LEN
      ENDIF
*
* IF input length < then output length
*   call routine to build code to pad output field with spaces
* ELSE
*   IF input length = Output length
*   Call routine to build an MVC instruction
*   This routine uses current source and target D(B)'s
*   and the output length to construct the instruction
* ELSE
*   input length > output length
*   Call routine to build an MVC instruction
*   This routine call will use the input length (since it shorter)
*   (source and target D(B)'s will be used
*   Build Code to check for truncation of only spaces
*   ENDIF
*   ENDIF
*   LH     R1,$BCB_ILEN              GET INPUT LEN
*   LH     R2,$BCB_OLEN              GET OUTPUT LEN
*
*   CR     R1,R2                     CHECK LENGTHS
*   BE     CHAR_CHAR_0050             EQUAL
*   BH     CHAR_CHAR_0100             I > O ->
*
* INPUT LENGTH LESS THAN OUTPUT -> NEED TO PAD
* Build Character padding code
*   #BAS   14,=A(SSP_0000)
*
* Build code TO MOVE CHARACTER FIELD TO CHARACTER FIELD
CHAR_CHAR_0050 DS 05
*   #BAS   14,=A(MVC_0000)           BURST MVC INSTRUCTION
*   B      CHAR_CHAR_0200

```

**Fig. 6f**

```

* INPUT field is too large to fit
* Build code TO MOVE CHARACTER FIELD TO CHARACTER FIELD using input field's length
CHAR_CHAR_0100 DS      0S
                LR      R1,R2
                #BAS     14,=A(MVC_0000)                BURST MVC INSTRUCTION
*
* MOVE CHECK FOR SPACES
* IF TRUNCATED DATA NOT SPACES THEN #@ERROR4

                IF (-NO_BUILD)
*
                MVC      0(CHAR_CHAR_010_L,R6),CHAR_CHAR_010
*
* SET LENGTH OF COMPARE
                LH       R7,$BCB_ILEN
                SR       R7,R1
                BCTR     R7,0
                STC      R7,CHAR_CHAR_010_OLEN_A(,R6)
*
* SET SOURCE DB TO SOURCE + OLEN-1
                LH       R7,WB_SOURCE_DB
                LA       R7,0(R1,R7)
                BCTR     R7,0
                STH      R7,CHAR_CHAR_010_SDBN_A(,R6)
*
* UPDATE BUFFER OFFSET
                LH       R7,WB_BUFFER255_OFFSET
                O        R7,=X'0000C000'
                STH      R7,CHAR_CHAR_010_B255_A(,R6)
*
* UPDATE #@ERROR4 BRANCH
                LH       R7,WB_CNVERR4_OFFSET
                STH      R7,CHAR_CHAR_010_BERR_A(,R6)
*
                ENDIF                (NO_BUILD)
*
                LA       R5,CHAR_CHAR_010_L(,R5)
                LA       R6,CHAR_CHAR_010_L(,R6)
*

```

Fig. 6g

```

* CHECK FOR TRANSLATION of CCSID's
* If translation requested call translation routine generator
* *** note translation routine will perform accumulation
* operation if API requested it. If accumulation is performed
* by the routine the IN_BCB (copy of API block used by generator)
* will be updated to turn off accumulation by the main process
* done upon CHARACTER subroutine (see above)
CHAR_CHAR_0200 DS      OS
      IF (TM,$BCB_PFLAG2,$BCB_CCSID_CNV,NZ)
*       IF IREG =2 AND SRC_ACCUM TR INST WILL BUMP REG
          SETF  SAVE_R2
          IF (CLC,$BCB_IREG,EQ,=H'2'),AND,
          (TM,$BCB_PFLAG1,$BCB_TRG_ACUM+$BCB_TRG_L_ACUM,NZ)
          RESETF SAVE_R2
          NI      $BCB_PFLAG1,X'FF'-$BCB_SRC_ACUM
          ENDIF
          RESETF XLATE_TO_E
          IF (TM,$BCB_PFLAG2,$BCB_CCSID_CNV_ATOE,O)
          SETF XLATE_TO_E
          ENDIF
          #BAS    14,=A(DO_XTAB_SHORT)
      ENDIF
*
CHAR_9999 DS      OS
      B      CHARACTER_END
*-----
* BURST CHARACTER TO CHARACTER ILEN > OLEN
* TEMPLATE CODE USED FOR NON-SPACE TRUNCATION
*-----
CHAR_CHAR_010 DS OS
CHAR_CHAR_010_OLEN_A EQU *-CHAR_CHAR_010+1    LEN OF CLC
CHAR_CHAR_010_SDBN_A EQU *-CHAR_CHAR_010+2    LOC OF SOURCE TO COMP
CHAR_CHAR_010_B255_A EQU *-CHAR_CHAR_010+4    LOC OF 255 SPACES
      CLC      0(0,0),0(0)                    SDB+(OLEN-1),BWK_BUFF255
CHAR_CHAR_010_BERR_A EQU *-CHAR_CHAR_010+2
      BNE      0(R12)                          NOT SPACES? -> #@ERROR4
CHAR_CHAR_010_L      EQU *-CHAR_CHAR_010
*-----

```

Fig. 6h



**DECLARATION AND POWER OF ATTORNEY**

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below next to my name.

I believe I am an original, first and joint inventor of the subject matter which is claimed and for which a patent is sought on the invention entitled **SYSTEM FOR GENERATING OPTIMIZED COMPUTER DATA FIELD CONVERSION ROUTINES**, the specification of which is filed herewith.

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims.

I acknowledge the duty to disclose information which is material to the examination of this application in accordance with Title 37, Code of Federal Regulations, § 1.56(a).

I hereby claim foreign priority benefits under Title 35, United States Code, § 119 of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application(s) for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

**PRIOR FOREIGN APPLICATION(S)**

Number	Country filed	Day/month/year	Priority Claimed Under 35 USC § 119
<hr/>			

I hereby claim the benefit under Title 35, United States Code, §§ 119(e) of any United States provisional application(s) listed below:

**PRIOR PROVISIONAL APPLICATION(S)**

Application Number	Filing Date
<hr/>	

I hereby claim the benefit under Title 35, United States Code § 120, of any United States application(s) listed below or under § 365(c) of any PCT international application(s) designating the United States of America listed below, and insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States or PCT International application(s) in the manner provided by the first paragraph of 35 U.S.C. 112, I acknowledge the duty to disclose information which is material to patentability as defined in 37 CFR 1.56 which became available between the filing date of the prior application and the national or PCT international filing date of this application.

**PARENT APPLICATION(S)**

U.S. Parent or PCT Parent Application Number	Parent Filing Date	Parent Patent Number (if applicable)
---	-----------------------	---

---

And I hereby appoint James David Jacobs (Reg. No. 24,299), Jonathan S. Caplan (Reg. No. 38,094), Chris Kolefas (Reg. No. 35,226), Victor DeVito (Reg. No. 36,325) and Harry K. Ahn (Reg. No. 40,243) my attorneys with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected therewith.